Software Development Principles

# Lecture 5
# Objects 1

**Lecturer:**

**Karen Nolan**

karen.nolan@it-tallaght.ie

# Topics

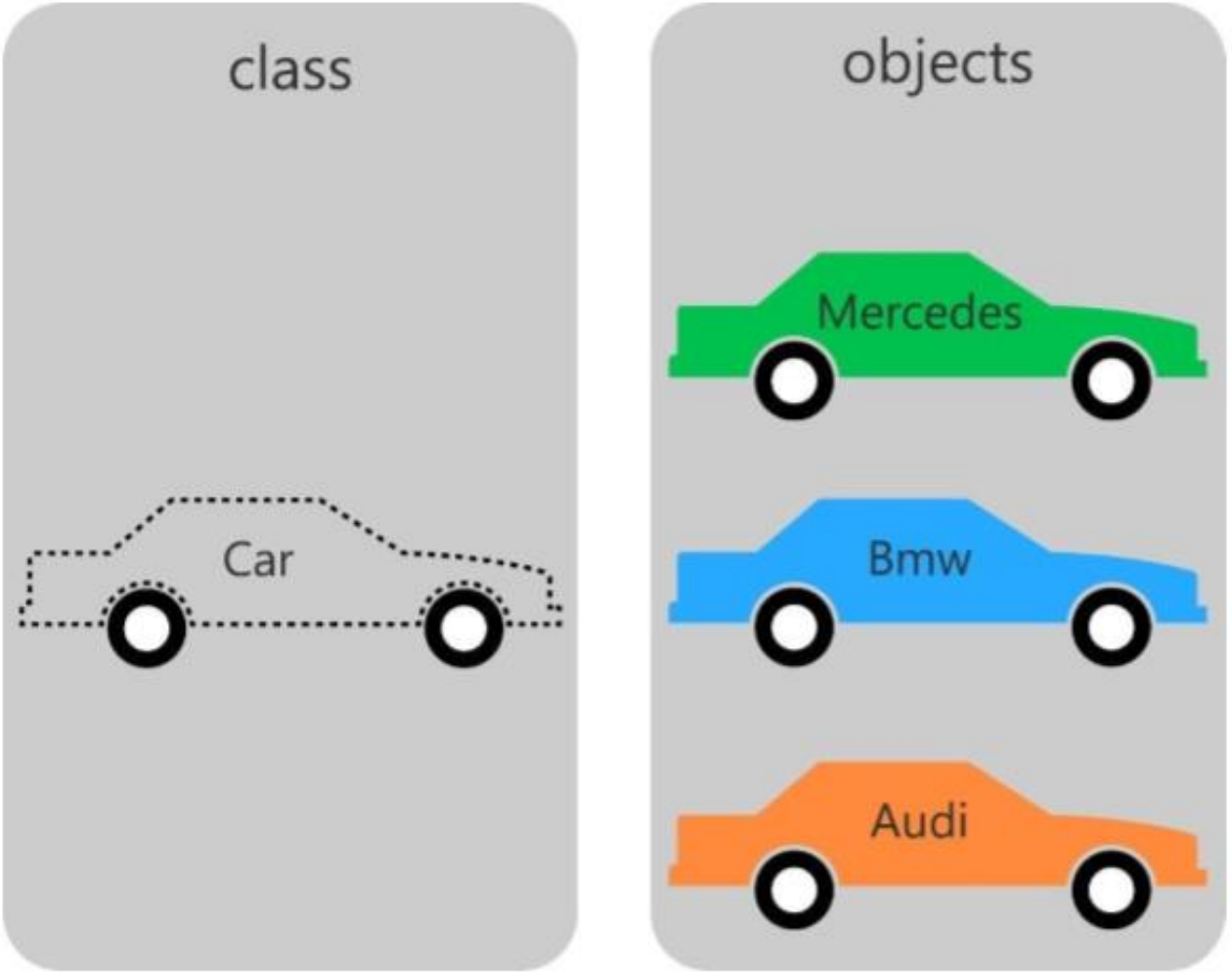
- Creating Basic Objects
- Class variables
- Initialization
- Print definitions

# Objects

- Create a class
- Then create an instance of a class
- Create a separate instance of the class
- Test both printing out attributes (class variables)
- Lets look at a car example….

# Objects - Car

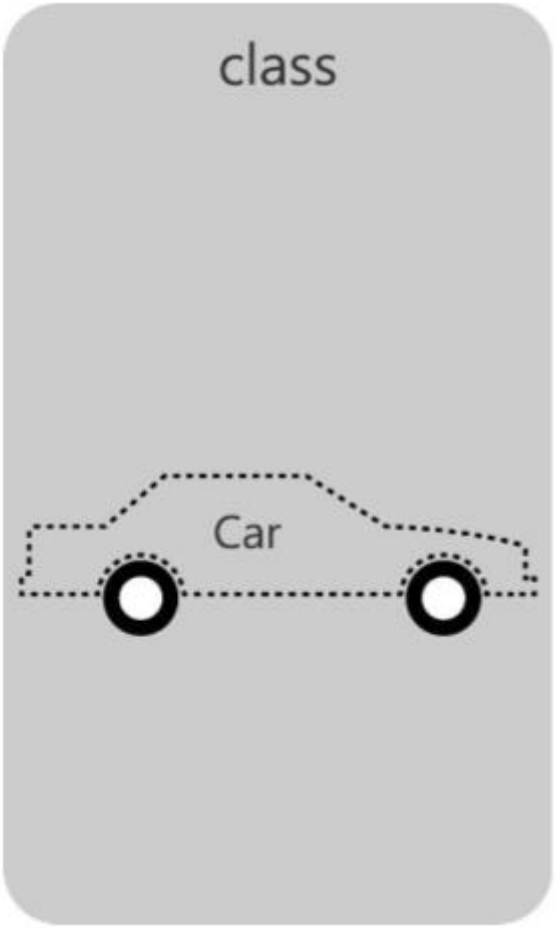
- Car



# Objects – Car Class

- Format of a class

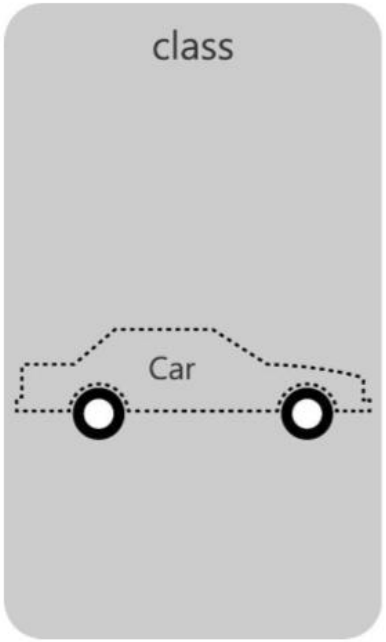
```
class Car:

      # Attributes

      # Definitions
```



# Objects – Car Class

- Format of a class (using class variables to begin with)

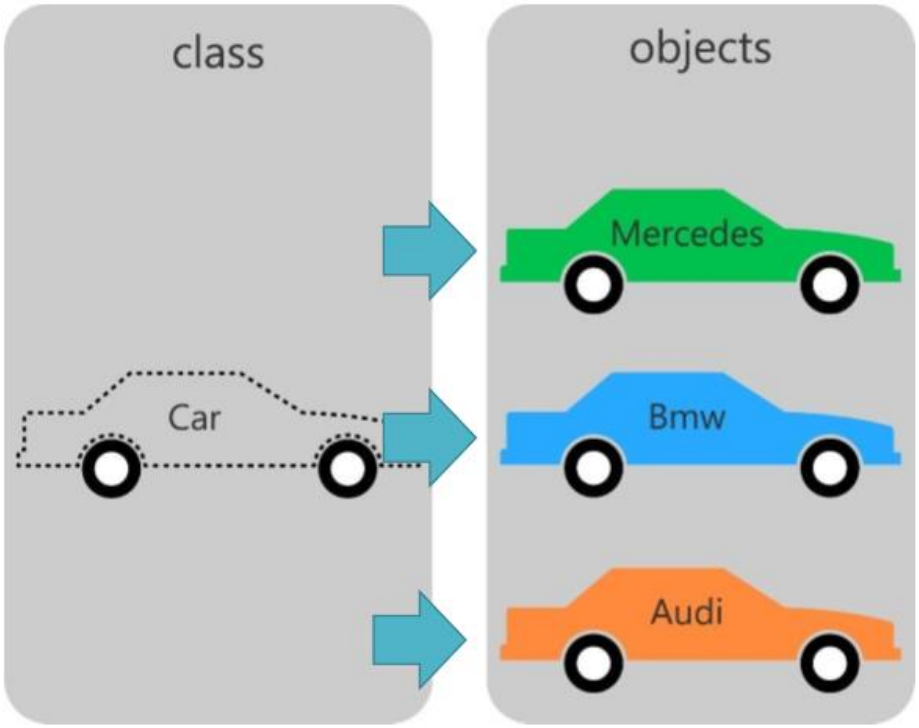
```
class Car:

      # Attributes first
      make = "unknown"
      model = "unknown"
      engineCC = 0.0

      # Definitions
```




- We are giving the variables a default value for all car instances, later we will use an alternative method

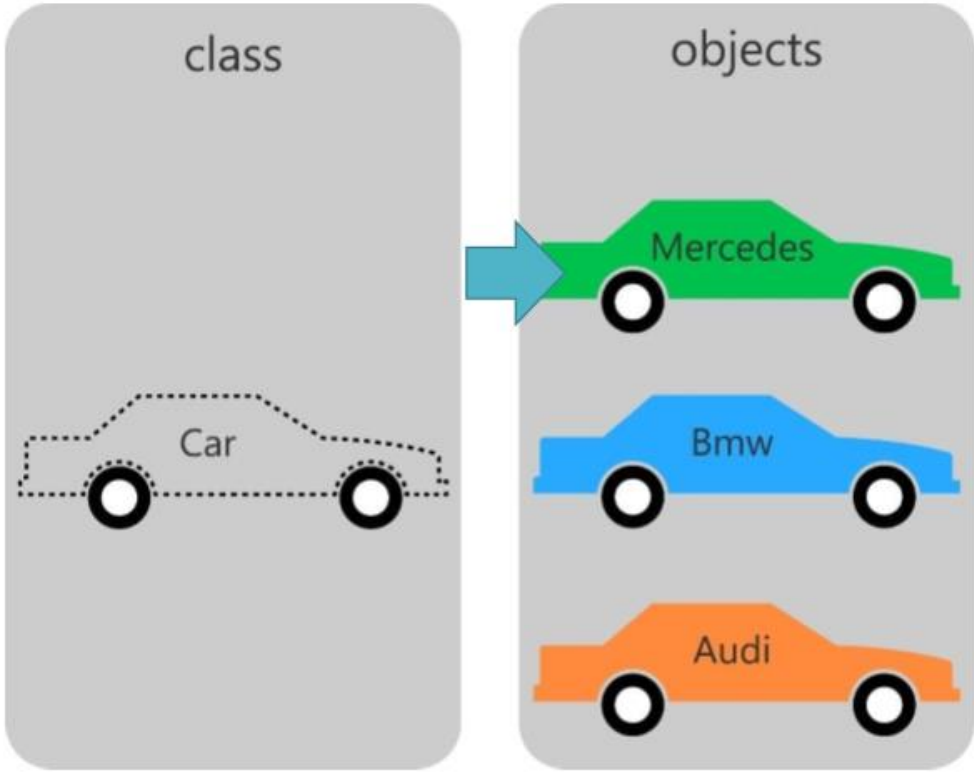
# Objects – Instance of a Car

- Next step is to create an instance of the car class
- This takes the class (blueprint or DNA of an object) and creates a copy of this class in memory (an instance)
- The instance gets all of the attributes and definitions of the class.



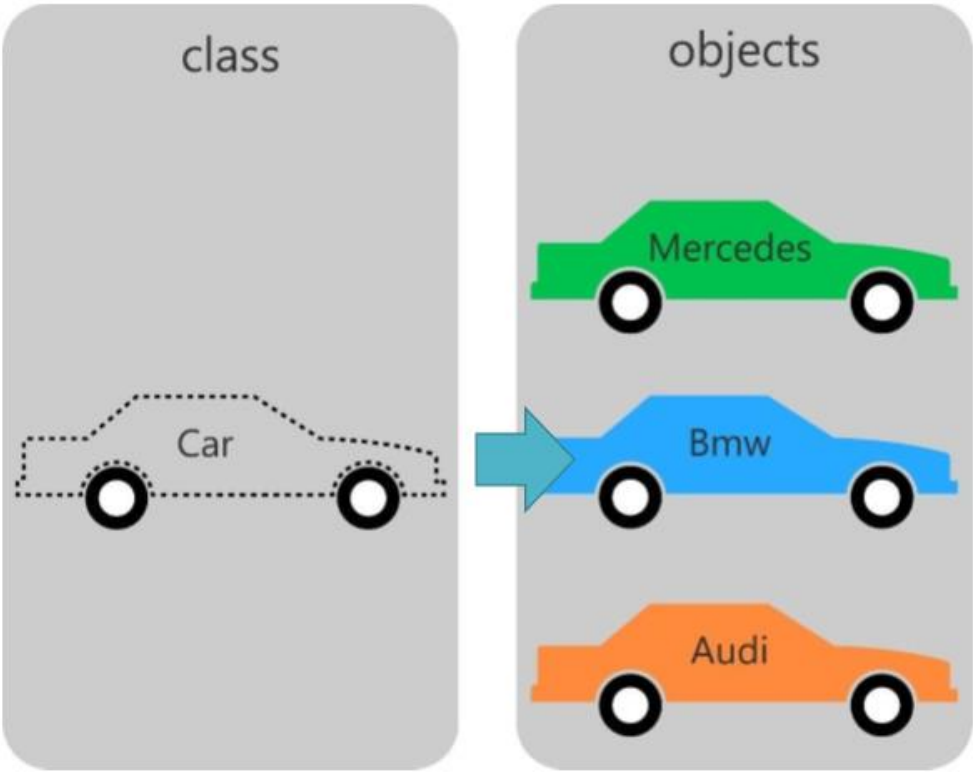
# Objects – Instance of a Car

```
class Car:

      # Attributes first
      make = "unknown"
      model = "unknown"
      engineCC = 0.0

      # Definitions



mercedes = Car()
bmw = Car()
audi = Car()
```



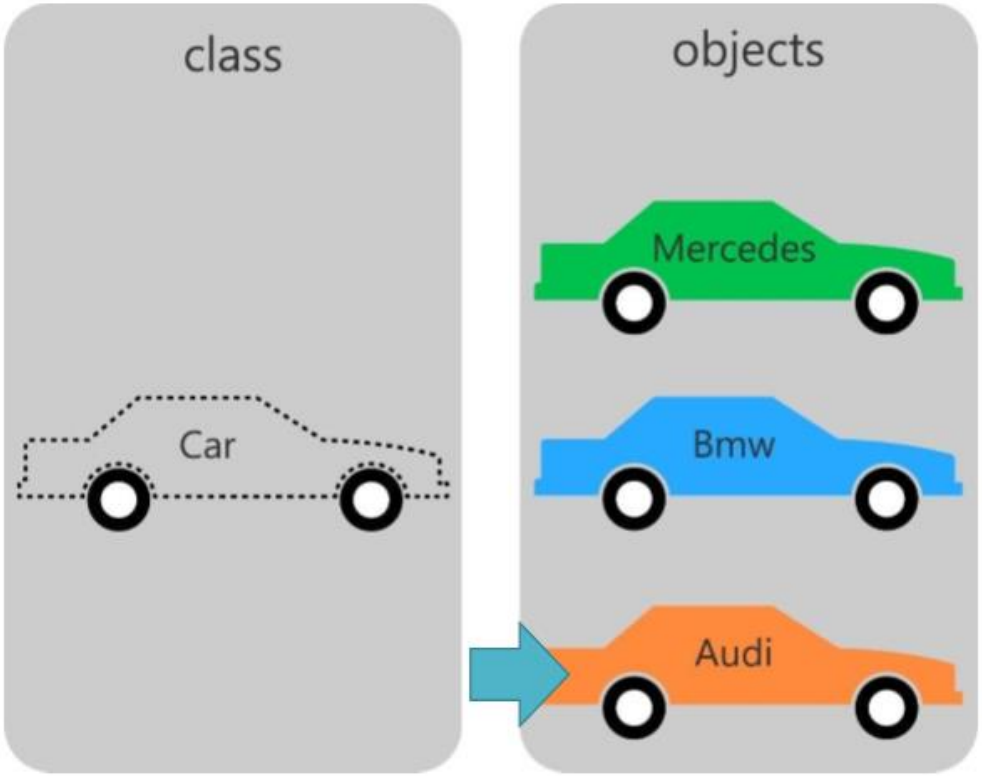
# Objects – Instance of a Car

```
class Car:

      # Attributes first
      make = "unknown"
      model = "unknown"
      engineCC = 0.0

      # Definitions



mercedes = Car()
bmw = Car()
audi = Car()
```



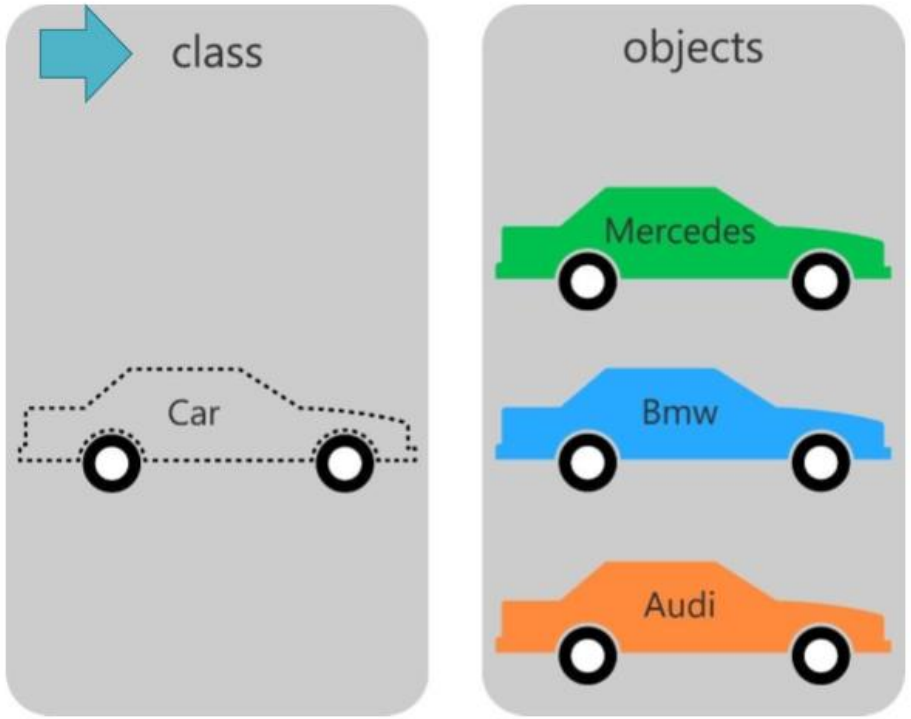
# Objects – Instance of a Car

```
class Car:

      # Attributes first
      make = "unknown"
      model = "unknown"
      engineCC = 0.0

      # Definitions



mercedes = Car()
bmw = Car()
audi = Car()
```



# Objects – Instance of a Car

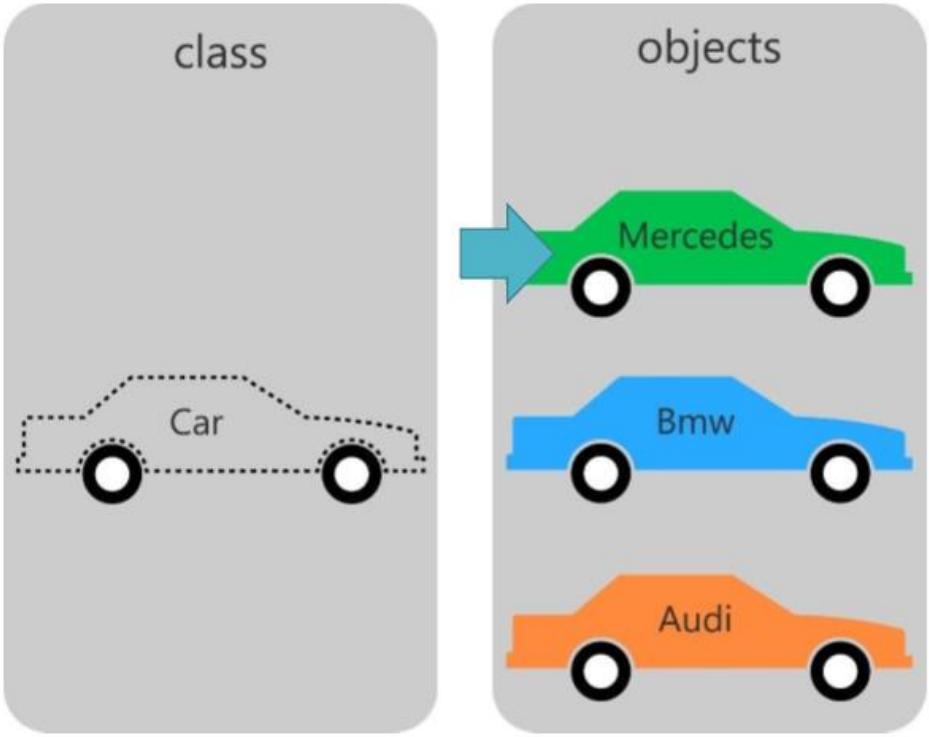
```
mercedes = Car()
bmw = Car()
audi = Car()

print(Car)
print(mercedes)
print(bmw)
print(audi)
```




```
<class '__main__.Car’>
<__main__.Car object at 0x029636D0>
<__main__.Car object at 0x029636F0>
<__main__.Car object at 0x02963750>
```

# Objects – Instance of a Car

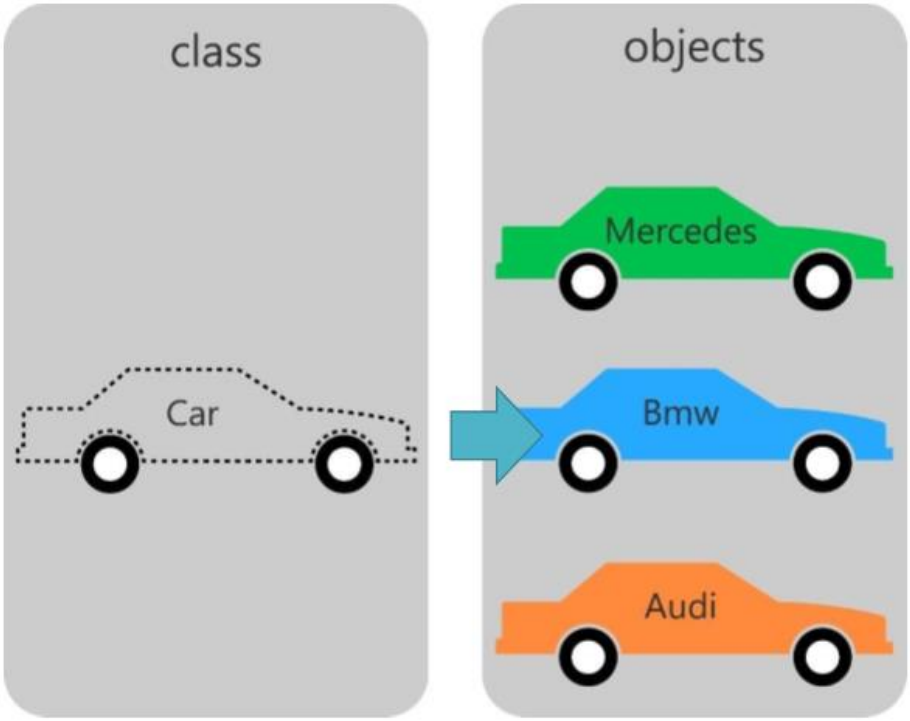
```
mercedes = Car()
bmw = Car()
audi = Car()

print(Car)
print(mercedes)
print(bmw)
print(audi)
```




```
<class '__main__.Car’>
<__main__.Car object at 0x029636D0>
<__main__.Car object at 0x029636F0>
<__main__.Car object at 0x02963750>
```

# Objects – Instance of a Car

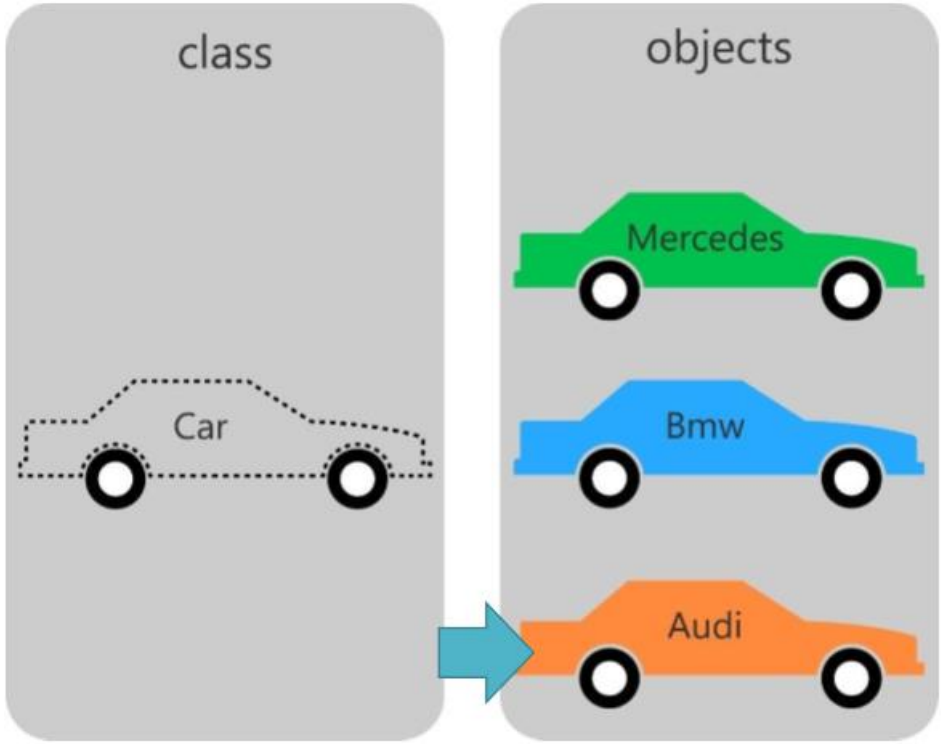
```
mercedes = Car()
bmw = Car()
audi = Car()

print(Car)
print(mercedes)
print(bmw)
print(audi)
```




```
<class '__main__.Car’>
<__main__.Car object at 0x029636D0>
<__main__.Car object at 0x029636F0>
<__main__.Car object at 0x02963750>
```

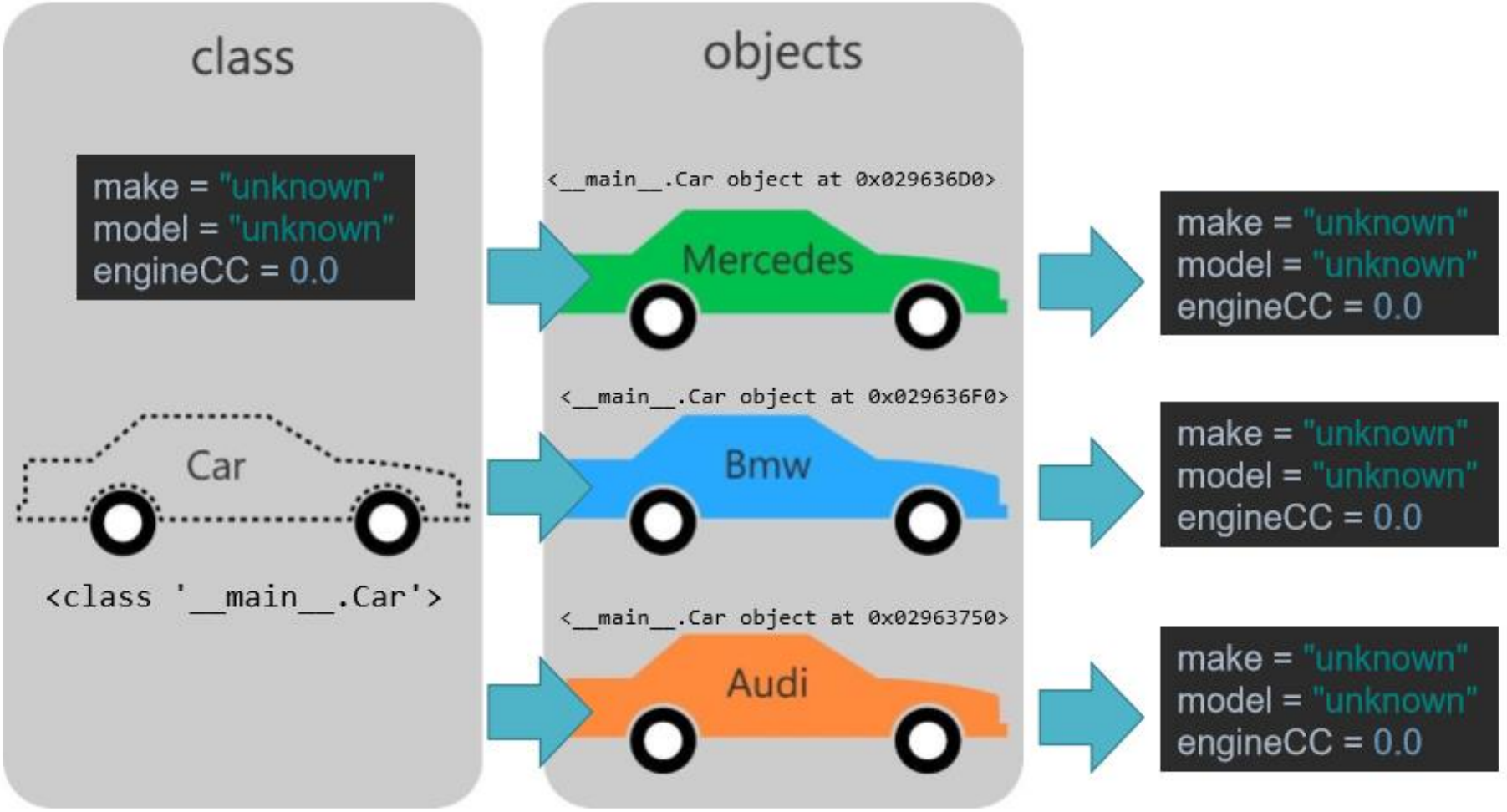
# Objects – Instance of a Car

```
mercedes = Car()
bmw = Car()
audi = Car()

print(Car)
print(mercedes)
print(bmw)
print(audi)
```

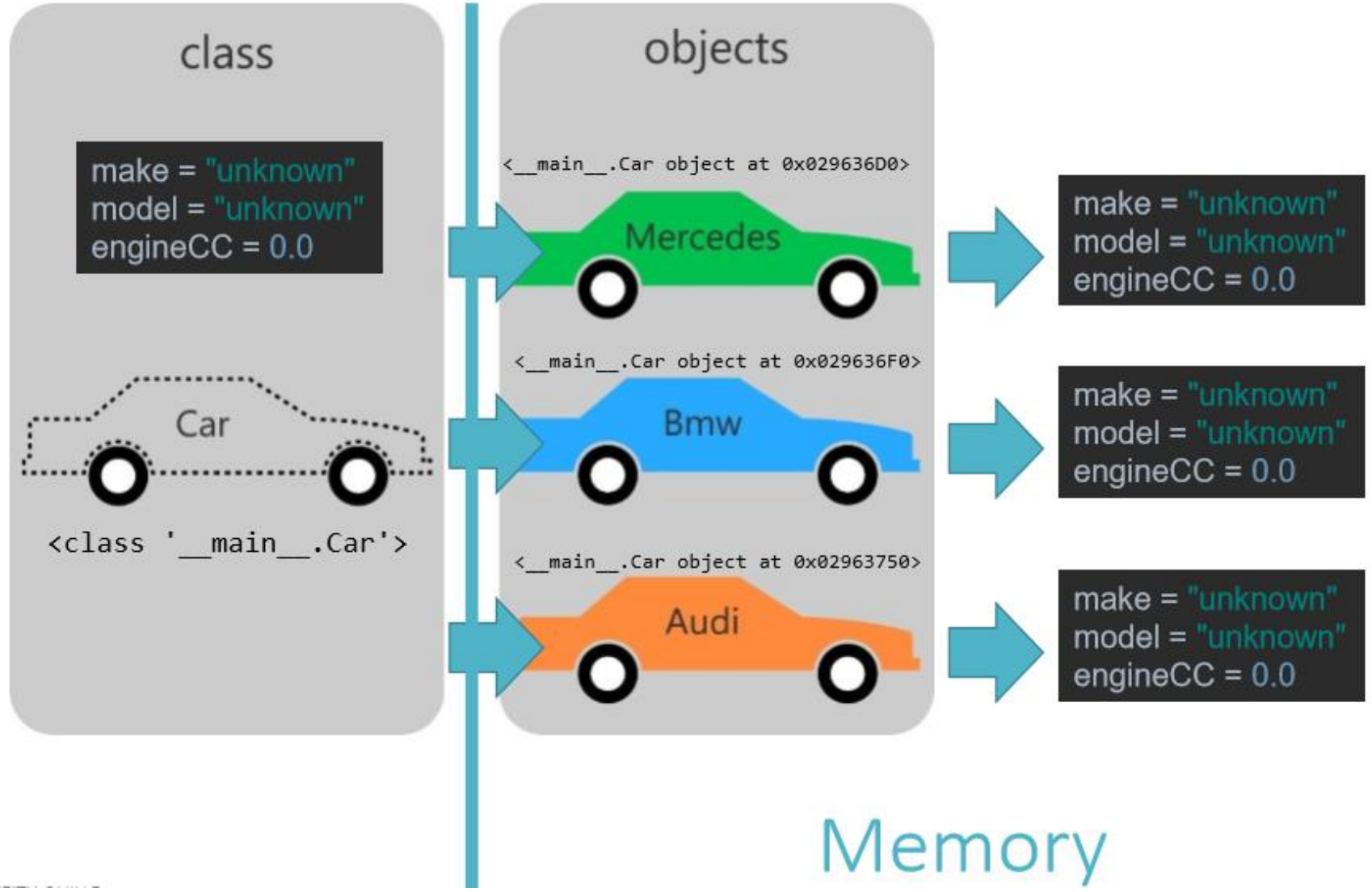



```
<class '__main__.Car’>
<__main__.Car object at 0x029636D0>
<__main__.Car object at 0x029636F0>
<__main__.Car object at 0x02963750>
```

# Objects – Instance of a Car



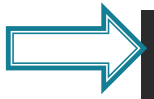
# Objects – Instance of a Car



# Objects – Instance of a Car

- Lets just look at the Mercedes instance.

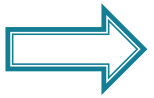
- **Step 1:**
  - An Instance of a Car is created.
  - Instance Name: **mercedes**



```
mercedes = Car()
print(mercedes.make)
print(mercedes.model)
print(mercedes.engineCC)

mercedes.make = "Mercedes"
mercedes.model = "C-Class"
mercedes.engineCC = 1600

print(mercedes.make)
print(mercedes.model)
print(mercedes.engineCC)
```

# Objects – Instance of a Car

- Lets just look at the Mercedes instance.



```
mercedes = Car()
print(mercedes.make)
print(mercedes.model)
print(mercedes.engineCC)

mercedes.make = "Mercedes"
mercedes.model = "C-Class"
mercedes.engineCC = 1600

print(mercedes.make)
print(mercedes.model)
print(mercedes.engineCC)
```

- **Step 2:**
  - Printing values in mercedes instance.
  - Using the:

`<instance name>.<attribute name>`

to access the instance attributes

# Objects – Instance of a Car

- Lets just look at the Mercedes instance.

```
mercedes = Car()
print(mercedes.make)
print(mercedes.model)
print(mercedes.engineCC)

mercedes.make = "Mercedes"
mercedes.model = "C-Class"
mercedes.engineCC = 1600

print(mercedes.make)
print(mercedes.model)
print(mercedes.engineCC)
```

- **Step 3:**
  - Updating / modifying values in mercedes instance.
  - Using the:
    `<instance name>.<attribute name> = new value`
    to access and modify the instance attributes

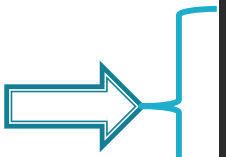
# Objects – Instance of a Car

- Lets just look at the Mercedes instance.

- **Step 4:**
  - Printing values in mercedes instance.
  - Using the:

`<instance name>.<attribute name>`

to access the instance attributes



```
mercedes = Car()
print(mercedes.make)
print(mercedes.model)
print(mercedes.engineCC)

mercedes.make = "Mercedes"
mercedes.model = "C-Class"
mercedes.engineCC = 1600

print(mercedes.make)
print(mercedes.model)
print(mercedes.engineCC)
```

# Objects: Example 1

- Create a class that represents an employee
- The attributes should be:
  - Name
  - EmployeeId
  - Department Salary
- Create an employee class, then create two instances of the class. Set all attributes in the instances to values of your choice. Print out each attribute value in each instance

# Objects: Example 2

- Create a class that represents an rectangle
- The attributes should be:
  - width
  - length
- Create a rectangle class, then create two instances of the class. Set all attributes in the instances to values of your choice. Print out each attribute value in each instance.
- Print the area of each rectangle

# Objects Definitions

- Along with attributes, objects may have definitions. These definitions are intimately tided to the data of the class.
- The previous class are really just states, as they have variables that have values or states.
- To add behaviour to the classes (and therefore the instances), we use definitions. There is a subtle difference between our previous definitions and definitions for classes that will be called as instances.
- Lets look at our first behaviour for the Car, Employee and rectangle class.

# Objects: Car Definitions

- Our car class, was useful, but it was very time consuming to print each attribute. Thus create a definition to do this work for us

```
class Car:

    # Attributes first
    make = "unknown"
    model = "unknown"
    engineCC = 0.0

    # Definitions
    def my_print(self):
        print("************************************")
        print("           car Details              ")
        print("************************************")
        print("Car Make          :", self.make)
        print("Car Model         :", self.model)
        print("Engine Size (cc) :", self.engineCC)
```

# Objects: Car Definitions

- This definition has:
  - No Arguments
  - No Returns

- Has a special argument `self`
  - This arguments refers to the current instance and not the class.

- We use the `self.make` to access the instance variables (in this case make)

```
# Definitions
def my_print(self):
      print("************************************")
      print("            car Details             ")
      print("************************************")
      print("Car Make          :", self.make)
      print("Car Model         :", self.model)
      print("Engine Size (cc) :", self.engineCC)
```

# Objects: Car Definitions

- Using the definition

```
class Car:

      # Attributes first
      make = "unknown"
      model = "unknown"
      engineCC = 0.0

      # Definitions
      def my_print(self):
            print("*********************************")
            print("           car Details           ")
            print("*********************************")
            print("Car Make          :", self.make)
            print("Car Model         :", self.model)
            print("Engine Size (cc) :", self.engineCC)
```

```
myCar = Car()

myCar.name = "Honda"
myCar.employeeId = " CRX"
myCar.department = 1600

myCar.my_print()
```

- Using the my_print definition of the car class, will save significant time, as its reuse of code saves multiple lines of typing

# Objects: Example 3

- Create a print definition for the Employee Class, and the Rectangle class.
- Test both, by creating an instance and testing the print() definition

# Objects: Example 4

- Create a print definition for the Rectangle class the prints its area….